Energy Efficient Target Tracking in Wireless Sensor Networks with Limited Sensing Range

Oualid Demigha *, Hamza Ould Slimane*, Abderahim Bouziani* and Walid-Khaled Hidouci †
*Ecole Militaire Polytechnique
Bordj El-Bahri, Algiers 16111. Algeria
Email: o_demigha@esi.dz, hamzabydos@gmail.com, abdou.bouzbouz@gmail.com
†Ecole Nationale Superieure d'Informatique
Oued-Smar, Algiers. Algeria
Email: w_hidouci@esi.dz

***Abstract*—In this paper, we propose a dynamic clustering protocol coupled with a consensus-based Kalman filter algorithm to self-organize Wireless Sensor Networks for localized tracking of a single moving target. Our proposed scheme takes opportunity from the fact that the target presence is a localized event. Therefore, We consider a WSN with limited sensing range to design a target tracking scheme using low-cost limited-energy nodes. Simulation results show a clear improvement in the network energy consumption, however state estimation quality degrades slightly compared with centralized approaches and other tracking schemes with limited sensing range that do not limit the set of tracking nodes. Our tracking scheme reduces the number of tasking nodes which reduces the network energy consumption.**

***Index Terms*—Energy conservation, dynamic clustering, localized target tracking, Kalman consensus filter, limited sensing range.**

I. INTRODUCTION

Wireless Sensor Network (WSN) is an emerging technology that consists of hundreds or thousands of tiny low-cost energy-limited nodes that have small capacities of sensing, processing and communication via radio medium. Typically, these nodes report captured data to a base station for further processing. They are equipped with a low-cost small-capacity batteries which are, in most cases, non-rechargeable and irreplaceable. Therefore, network lifetime is considered as an important issue for many key applications such as: target tracking [1].

In contrast to high-cost sophisticated surveillance technologies, WSNs use cheap technology that do not rely on any centralized infrastructure.

This technology which aims at providing the same performance as do traditional systems, brings up new challenges related to data processing algorithms, communication systems and network organization. In many cases, collaboration among nodes helps at solving these challenging open-issues.

In contrast to single-node tracking systems, collaborative target tracking fuses data transmitted by many nodes and produces state-estimation of the target. However, these measurements are noisy, redundant and non-synchronized and the inter-node communication is an energy-consuming task. Furthermore, neither reliable communication protocols nor complex data processing algorithms can be implemented on a sensor node because of its limited processing and communication capacities.

Therefore, energy efficiency in target tracking is a key issue in WSN and it can be achieved using different methods [2]. One of them is the prediction-based schemes coupled with selective activation. Sensor nodes can collaboratively generate predictions of the target location by executing an *in-network light-weight* data fusion algorithm. The gain of such algorithms is two-fold: (i) it generates state-estimates of the target, and (ii) it produces state-predictions for the next sampling period which are used to activate selected nodes (implicitly or explicitly by sending an activation message).

In many cases, this method requires collaboration among nodes to provide accurate data in presence of noisy sensor measurements transmitted over noisy communication links. Furthermore, sensor readings from the low-cost limited sensing range components are, in fact, less accurate but they are close to the target which, in turn, can be detected by more than one sensor at the same time. Hence, the sensor network can take profit form this data redundancy to improve the tracking quality.

In this paper, we consider a WSN with limited sensing range that *localizes* the data fusion algorithm within the target detection zone and *self-organizes* nodes within dynamic clusters that move along the target trajectory. It is interesting to study this type of WSN because they help at minimizing the network energy consumption and provide acceptable tracking quality by selecting the appropriate tasking nodes.

This paper is organized as follows: in Section II, we give some definitions and system models. In Section III, we describe some related works proposed in the literature to reduce energy consumption via distributed Kalman filter algorithm. In Section IV, we present our proposed method that consists of two main components: (1) the Kalman consensus filter and (2) the dynamic clustering protocol. In Section V, we discuss the simulation results and the tradeoff between the energy consumption and the estimation quality. Finally, in Section VI, we conclude the paper.

 ISBN: 978-1-61208-166-3

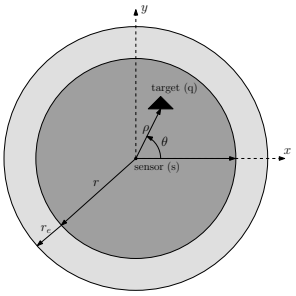

Fig. 1. Probabilistic detection model

II. BACKGROUND

In the following subsections, we give some basic definitions of the WSN model and the centralized Kalman filter. After that we describe the mathematical model of the distributed Kalman filter adopted in our proposed method.

A. System Model and Assumptions

We model the wireless sensor network by a non-oriented graph $G(V, E)$ where $V = \{s_1, s_2, \cdots, s_n\}$ is the set of nodes and $E = \{(s_i, s_j) \mid \|s_i - s_j\| \leq R_c\}$ is the set of communication links. R_c is the communication range of each node and $\|s_i - s_j\|$ is the Euclidean distance between nodes s_i and s_j. We assume that the sensing range R_s is uniform among all the nodes and it verifies the condition required for coverage and connectivity constraints, i.e., $R_c \geq 2R_s$.

We suppose that the target state is a 4-tuple vector:

$$X = (x, y, \dot{x}, \dot{y}) \in R^4$$

where (x, y) and $(\dot{x}, \dot{y})$ are respectively, the target position coordinates and its velocity along X and Y axes. Each node measures the distance to the target ρ and the angle between the X axis and the target position vector θ. For target detection, we use a probabilistic model expressed by the equation 1:

$$p_s(q) = \begin{cases} 0 & \text{if } r + r_e \leq \|s - q\| \\ e^{-\alpha(\|s-q\|-(r-r_e))^\beta} & \text{if } r - r_e \leq \|s - q\| \leq r + r_e \\ 1 & \text{if } r - r_e \geq \|s - q\| \end{cases} \quad (1)$$

where $\|s - q\|$ is the Euclidean distance between sensor s target q, r is the sensing range of s and r_e is the sensing error ($r_e \ll r$). α and β are constants (see Figure 1).

We assume also that nodes are initially in the sleep state which guarantees minimum energy consumption. In fact, in this state, all the nodes' hardware units are OFF, except the processing unit and a low-power *paging channel* to receive *wake-up* messages. Upon receiving a wake-up message, nodes start up all their hardware units. Nodes are supposed aware of their geographic positions and each node maintains a list of its neighboring nodes.

The first target detection is supposed done successfully and the first activation is performed via an *external* activation message.

B. Centralized Kalman Filter

We assume that the target state and the measurement models are respectively defined by the following linear equations:

$$x_{k+1} = A_k x_k + B_k u_k + w_k$$
$$z_k = H_k x_k + v_k$$

where: A is the matrix that relates the previous target state to the current one, B is the matrix that relates commands to the current target state, w_k is the system noise, H is the matrix that relates the measurements to the current target state and v_k is the measurements' noise at time step k. x_k is the target state vector at time step k.

We suppose that w and v are white noises with Q and R covariances respectively: $p(w) \sim N(0, Q)$ and $p(v) \sim N(0, R)$. Also, we suppose that matrices A and H are detectable and all matrices A, B, H, Q and R are time-independent. For *Constant-Velocity* target model, matrix A_k and H_k have these values:

$$A_k = A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, H_k = H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

We denote $\hat{x}_k^-$ and $\hat{x}_k$ as the *a priori* and the *a posteriori* target state estimations at the time step k, respectively.

As same, the a priori and a posteriori estimation error covariance matrices P_k^-, P_k at time step k are defined by:

$$P_k^- = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T]$$
$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]$$

The Kalman gain factor at time step k is defined by:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

We summarize the Kalman model by the following recursive steps:

1) Prediction Step:
   - Next step state prediction:
   $$x_{k+1}^- = A_k \hat{x}_k + B_k u_k \quad (2)$$
   - Next step error covariance prediction:
   $$P_{k+1}^- = A_k P_k A_k^T + Q_k \quad (3)$$
2) Update Step:
   - Kalman gain:
   $$K_k = P_k^- H_k^T (H P_k^- H^T + R)^{-1} \quad (4)$$
   - Estimation update:
   $$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H_k x_k^-) \quad (5)$$
   - Error covariance update:
   $$P_k = (I - K_k H_k) P_k^- \quad (6)$$

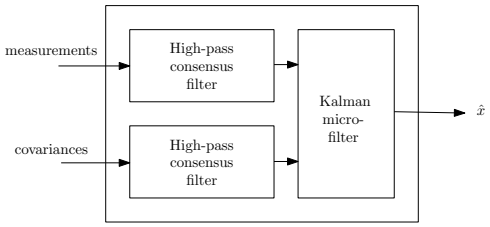

Fig. 2. Micro-filter architecture of a DKF node

C. Consensus on Kalman micro-filters

Decentralized Kalman filter gives target state estimation using a set of k micro-filters. This model requires *all-to-all* communications. In [3], the authors propose a Distributed Kalman Filter (DKF) that uses high-pass and low-pass filters to fuse data. It can fuse heterogeneous data obtained from non-linear sensing model:

$$y_i = h_i(x_k) + v_i^k$$

All the nodes have the same architecture as illustrated in Figure 2.

The system to be observed is modeled as follows:
$x_k = A_k x_{k-1} + B_k w_k$ and $y_i^k = H_i^k x_k + v_i^k$

In addition to the Kalman filter model equations described in Section II-B, two new values are included into the model namely: (1) the fusion of the error covariance inverse matrix and (2) the fusion of the measurements. These two values are expressed respectively by:

$$S_k = \frac{\sum_{i=1}^{n} H_i^{kT} R_i^{-1} H_i^k}{n}$$

and

$$y_k = \frac{\sum_{i=1}^{n} H_i^{kT} R_i^{-1} y_i^k}{n}$$

Each node executes the following calculations:

$$M_i^k = (P_{i,k}^{-1} + S_k)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + M_i^k (y_k - S_k \hat{x}_k^-)$$

$$P_i^{k+1} = A_k M_i^k A_k^T + B_k Q_i^k B_k^T$$

$$\hat{x}_{k+1} = A_k \hat{x}_k$$

with: $Q_i^k = nQ_k$ and $P_i^0 = nP_0$. The estimations are identical in all nodes, i.e.,

$$\hat{x}_i^k = \hat{x}_k, \forall i$$

The filter dynamic is expressed by equation 7 (for more details see [4]):

$$\begin{cases} \dot{q}_i = -\beta \hat{L} q_i - \beta \hat{L} u_i \\ p_i = q_i + u_i \end{cases} \quad (7)$$

where $\hat{L} = L \bigotimes I_m$ is the m-dimension graph laplacian, u_i is the node input, q_i is the Kalman filter state and β ($\beta > 0$) is the gain (it should be big enough for random deployed topologies). The filter output is computed according to equation 8:

$$\begin{cases} \dot{q}_i = \beta \sum_{j \in N_i} (q_j - q_i) + \beta \sum_{j \in N_i} (u_j - u_i) \quad \beta > 0 \\ y_i = q_i + u_i \end{cases} \quad (8)$$

With N_i is the i^{th} node's neighbors set (the reader could refer to [5] to learn more about filter discreatization in connected graphs).

III. Related Work

The Kalman Consensus Filter (KCF) is proposed in [3]. It uses a set of *k Kalman micro-filters* to fuse heterogeneous data received from sensors with non-linear sensing models. There are two variants of this approach: one fuses measurements and the other fuses estimations. In the measurements' fusion variant, low-pass and band-pass filters are modified into *high-gain* high-pass filters. The other variant fuses estimations instead of measurements in order to accelerate the consensus convergence. This filter uses latency-generating power-consuming complex matrix computations which may fail at detecting fast targets. Furthermore, the algorithm makes the assumption that all nodes can observe the target which may not hold all the time.

In [6], the authors use biparti graphs to distribute the Kalman Filter (KF) model. In this approach, the KF model is distributed on the whole network and the global model is decomposed into n_l ($n_l \ll n$ and n is the network size) reduced sub-models, each one is executed by a micro-filter in a single node. Each node computes its local estimation and fuses it with the received estimations. Biparti graphs are used when dependencies exist between these sub-models. This method is suitable for estimations' fusion because it includes data correlation between local estimations.

Distributed Kalman filter with Gossip communications is proposed in [7]. Each node can sense only a part of the observed phenomenon, i.e., each node can measure or estimate a subset of the target state attributes and communicates them to its neighbors and then deduces the missed attributes. There are two drawbacks to this method: (1) message communication complexity, i.e., nodes exchange many messages (estimations and error covariance matrix) , and (2) topology-dependency model, i.e., strong network connectivity is required for estimations' communication between neighboring nodes.

Olfati et al. propose a distributed Kalman filter with limited sensing range [8] in which nodes implement local micro-filters and reach a consensus using message passing communication model. This scheme makes the assumption that passive nodes (nodes that do not detect the target) are considered with *no contribution*. Even that, they are included in the fusion step.

Instead of sending long messages, authors of [9] propose that nodes send only *one bit* information. A *quantification function* is defined to represent node's estimation and the filter is then executed in two distinct procedures: an observation-transmission procedure and a reception-estimation procedure.

 ISBN: 978-1-61208-166-3

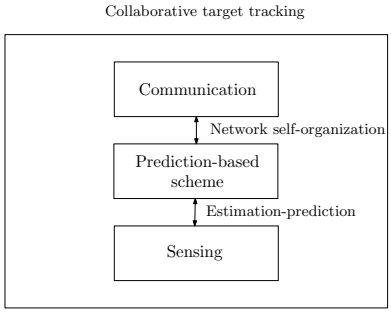

Fig. 3. Problem characterization

The main disadvantage of such approaches is that the quantification function may induce information loss when lossy links are present in the network.

IV. Proposed Method

All the above-mentioned approaches do not consider the problem of limiting the number of nodes participating in the tracking task and suppose that the target can be observed by the *whole* network. However, these two assumptions do not hold in all cases: i.e., in a 2D ground deployed WSN, only nodes that are close to the phenomenon can sense it; the other nodes can not. In addition, low-power nodes have limited sensing ranges and can communicate with a reduced set of neighbors.

This aspect can be exploited to reduce the energy consumption in a target tracking application. Figure 3 shows the relationship between the sensing component and the communication component in a sensor node that uses a prediction-based scheme. A *light-weight* estimation-prediction algorithm can be used to estimate the target state and predict its next position. This helps at waking-up the most appropriate nodes to track the target and at best organizing the network communications. Consequently, The other nodes remain in the *sleep* state which saves much more energy than in a *periodic sampling*-based target tracking scheme.

Indeed, periodic sampling provides more accurate data but it greatly reduces the energy resources of the nodes. Prediction-based schemes are more appropriate in a dense network where not all nodes are needed to be woken-up.

Therefore, two issues are to be considered here: (1) the estimation algorithm should be distributed over a subset of nodes that are close to the target, and (2) the tracking group should be dynamic depending on the target dynamic model. To resolve these issues we propose a Distributed Kalman Filtering approach with Dynamic Clustering (DKF_DC).

Our method is inspired by the work in [8], but instead of tasking all the network nodes, it uses a dynamic clustering to limit messages exchanges between nodes participating in the estimation process. Our clustering protocol consists of two phases: (1) leader election phase and (2) cluster reconfiguration phase.

Leader election is executed among *active* nodes that are close to the target. The other nodes stay inactive to save their energy resources. Therefore, nodes wake-up only when they receive activation messages to adhere to the current cluster. Unlike centralized fusion methods, the cluster-head in our method is not considered as a fusion center but as a cluster manager that is responsible for its reorganization. Hence, communications are performed between all the active nodes and not only between the active nodes and the cluster-head.

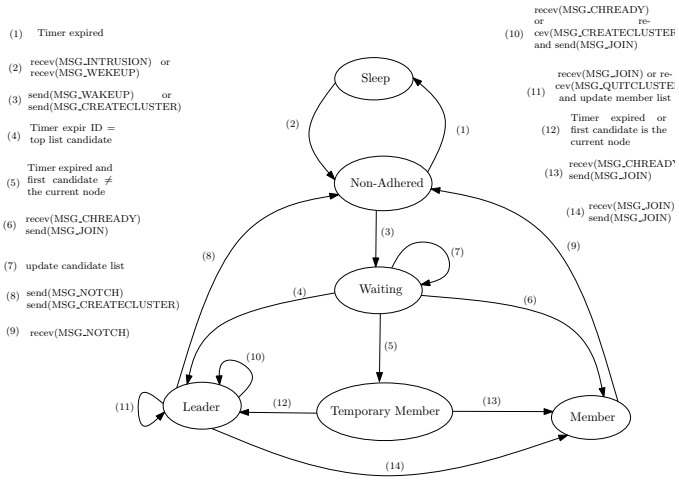

Fig. 4. State-transition diagram of the proposed clustering protocol

Continuous target tracking is guaranteed by allowing a subset of the last cluster members to adhere to the current cluster. That is what ensures the propagation of the estimation information along the target trajectory.

A. Cluster Formation and Leader Election

When a node receives an external intrusion message MSG_INTRUSION that contains the target state estimation recorded by some border nodes, it wakes-up and triggers the leader election phase. First, it sends a wake-up message MSG_WAKEUP to all its neighboring nodes, then it broadcasts a cluster creation message MSG_CREATECLUSTER that contains the first detected position.

Nodes that receive a MSG_CREATECLUSTER message compute a *local decision value* using measures like: (1) the distance between the node and the target, (2) the last estimation quality measured by the covariance matrix P_k issued from the Kalman filter or (3) the residual energy of the node.

As illustrated in the state-transition diagram in Figure 4, a node leaves the *non-adhered* state to go either back to the *sleep* state when a waiting timer expires or to the *waiting* state upon receiving a MSG_CREATECLUSTER message. The node within this state, computes the decision value and broadcasts it to its neighbors. If it receives a value sent from some neighboring node then it updates its *candidates' list* that contains couples (sender, value). Another timer is alarmed to wait for receiving such values. After its expiration, the waiting node decides to become a leader if it is the top list candidate. Otherwise, it becomes a *temporary member*. During the waiting time, if the node receives a MSG_CHREADY

 ISBN: 978-1-61208-166-3

message that contains the cluster-head ID, then it adheres as a member to this cluster.

Temporary-member node discards the top list candidate and waits for receiving a MSG_CHREADY message to adhere to that cluster. If it does not receive such message, it becomes leader if it is the top list candidate.

Similarly, a member node leaves this state upon receiving a MSG_NOTCH message sent by a leader. Consequently, it goes back to the non-adhered state.

B. Cluster Reconfiguration

The leader node checks the target state estimation to decide about the cluster reconfiguration. When it detects that the target is lost, then it performs the following two tasks:

1) Sending back a MSG_JOIN message to force the sender of a MSG_CREATECLUSTER or a MSG_CHREADY message to adhere to its cluster.
2) Updating the cluster members list upon receiving a MSG_JOIN or a MSG_QUITCLUSTER messages.

A leader leaves this state when one of the following events occurs:

- Receiving a MSG_JOIN message to become a member of the message sender's cluster.
- Losing the target: the cluster should be reconfigured and the nodes return back to the non-adhered state.

The cluster reconfiguration operation consists of updating the candidates' list and informing member nodes that the leadership has changed using a MSG_NOTCH message. Upon receiving this message, nodes trigger a new election process that may include previous members.

On the contrary to the scheme proposed in [10], our dynamic clustering protocol prevents multi-cluster tracking by letting only direct neighboring nodes to adhere to the cluster and force the other nodes to return back to the sleep state. After constructing the tracking cluster, the data fusion phase comes into play to generate target state estimation.

C. Target State Estimation

The member nodes of the current cluster perform the sampling to detect the target. They (including the leader node) compute their information matrix u_i and U_i as follows:

$$u_i = H_i^T R_i^{-1} z_i \tag{9}$$

$$U_i = H_i^T R_i^{-1} H_i \tag{10}$$

Equations 9 and 10 contain respectively the measurements information and the measurements errors information. The Kalman filter fuses estimation errors to generate updated state estimations. After that, each node broadcasts a message $m_i = \{u_i, U_i, \bar{x}_i\}$ containing the measurements, the measurements errors and the last state estimation $\bar{x}_i$ to all the cluster members. Each node waits then for receiving such messages from the other members to fuse the information matrix and the vectors y_i and S_i as follows: $y_i = \sum_{j \in J_i} u_j$ and $S_i = \sum_{j \in J_i} U_j$.

At the end of the data fusion phase, each node estimates the target state using KCF:

$$M_i = (P_i^{-1} + S_i)^{-1} \tag{11}$$

$$\hat{x}_i = \bar{x}_i + M_i(y_i - S_i\bar{x}_i) + \gamma M_i \sum_{j \in N_i} (\bar{x}_j - \bar{x}_i) \tag{12}$$

After that, nodes update their respective micro-filter states using equations 13 and 14:

$$P_i = AM_iA^T + BQB^T \tag{13}$$

$$\bar{x}_i = A\hat{x}_i \tag{14}$$

The leader checks the distance to the target and eventually the number of the active nodes in its cluster or the residual energy to decide about the cluster reconfiguration. Consequently, it updates its list of candidates and assigns the leader task to the most appropriate member.

V. Simulations and Results

We use TOSSIM [11] to validate our proposed method by simulation. TOSSIM is a discrete-time simulator of TinyOS operating system for wireless sensor motes. We compared our method with three different target tracking schemes discussed in Section III which are: (1) centralized Kalman filter (CKF) [12], (2) distributed Kalman filter with limited sensing range (DKF_LSR) [8], (3) distributed Kalman filter with gossip communications (DKF_GOSSIP) [7]. The CKF is considered as the base reference for our comparisons.

A. Simulation Setup

Simulation parameters that we vary are: (1) the sampling period, (2) the network size (or network density) and (3) the target velocity. The communication range is set to $50m$, the sensing range is set to $15m$ and the target model is Gauss-Markov.

The nodes' energy consumption is evaluated using POWER-TOSSIM and the estimation quality is measured by the mean square error between the real target position and estimated position:$\epsilon = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}$.

In the following subsections, we present the simulation results we have obtained regarding two metrics, namely: energy consumption and estimation quality.

B. Energy Consumption

First, we simulate a 100 nodes WSN that consists of randomly deployed on a 2D area of $200 \times 200m^2$ surface. By varying the sampling period within the set of values from $1s$ to $3s$ we obtain the graphs in the figure 5.

As shown in this figure, the network average energy consumption of the different simulated schemes is inversely proportional to the sampling period time because of the number of data messages exchanged between nodes. CKF and DKF_LSR consume much more energy than DKF_GOSSIP and our method DKF_DC because of the centralized nature of CKF and the non-limited number of nodes that participate in the target state estimation in DKF_LSR. The reduced network

 ISBN: 978-1-61208-166-3

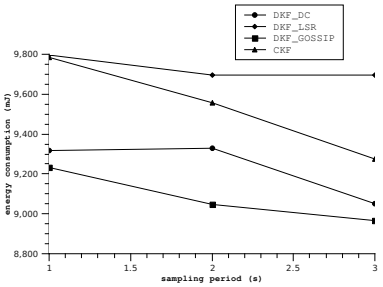

Fig. 5. Energy consumption of the network vs. Sampling period

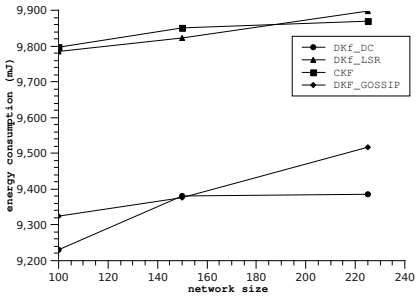

Fig. 6. Energy consumption of the network vs. Network density

energy consumption in DKF_DC is due to the fact that the dynamic clustering protocol limits the number of nodes involved in the tracking task.

We also evaluate the network energy consumption by varying the network size within $\{225, 150, 100\}$ and setting the deployment area surface to $200 \times 200m^2$. The sampling period is set to $1s$. We obtain the results in figure 6, in which we observe that the dense nature of a WSN influences the network energy consumption. In CKF and DKF_LSR methods, the number of tasking nodes is high which induces an excessive energy consumption because each node executes a sensing operation every tracking step. Sensing and communication energy consumption is high than computation consumption that is what explains why DKF_DC outperforms these two methods.

C. Estimation Quality

The estimation quality of CKF, DKF_DC and DKF_LSR schemes are evaluated for different sampling periods (see Figures 7, 8, 9).

As we can see in the three figures, the estimation quality of our method is less than those of CKF and DKF_LSR, and CKF outperforms all the other schemes in respect to the different sampling periods. In our method, the reduced set of participating nodes in the estimation process may decrease the total nodes' utility when less-appropriate nodes are chosen. Therefore, including all the network nodes in the estimation process and considering a uniform distributed noise model, this improves the estimation quality and convergence, because much data are fused despite the fact that they contribute or not in the estimation. Picks can be also observed on the graphs of

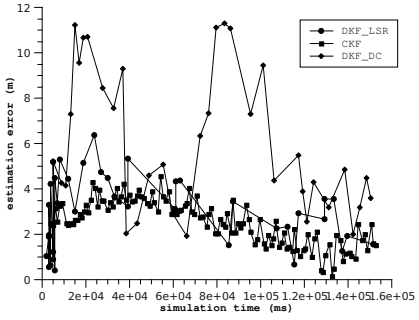

Fig. 7. Estimation quality for a $1s$ sampling period

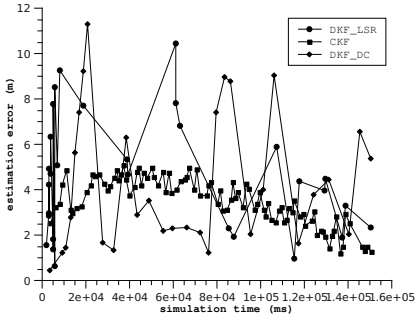

Fig. 8. Estimation quality for a $2s$ sampling period

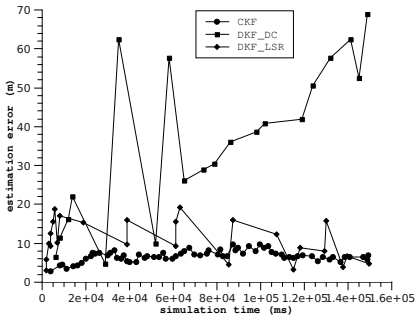

Fig. 9. Estimation quality for a $3s$ sampling period

DKF_DC due to the cluster reconfiguration process. Figure 9 shows that with a large sampling period, our method presents poor estimation quality because of the latency generated by the clustering protocol. This can be considered as a drawback of DKF_DC.

Finally, the estimation quality of CKF, DKF_LSR and DKF_DC schemes when varying the target speed within $\{1m/s, 2m/s, 4m/s\}$ is presented in Figures: 10, 11, 13, respectively.

In figure 10, we can see that the estimation error of CKF decreases for different target speeds. Indeed, we realize that the Kalman filter presents some picks in the beginning of the estimation process, but they disappear after that and the estimation error converges to zero due to the recursive nature of KF. Figure 11 shows that for targets with relatively low speeds, the estimation error of DKF_DC converges to zero. However, when the target speeds up, it overcomes the cluster

 ISBN: 978-1-61208-166-3

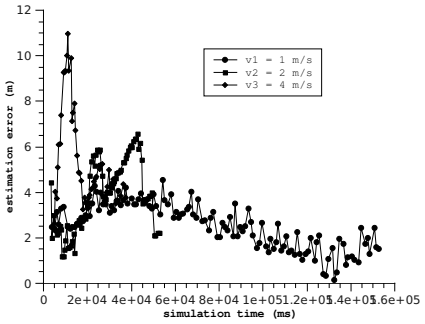

Fig. 10. CKF estimation quality vs. Target velocity

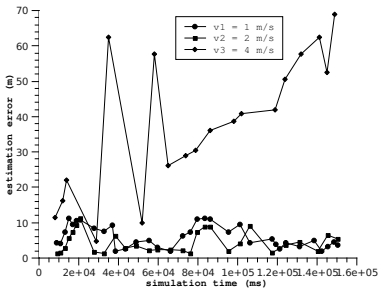

Fig. 11. DKF_DC estimation quality vs. Target velocity

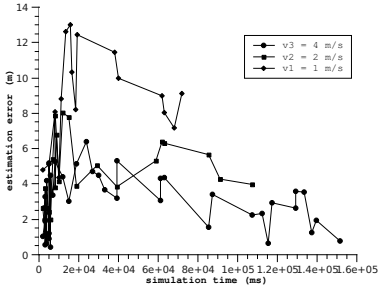

Fig. 12. DKF_LSR estimation quality vs. Target velocity

reconfiguration process. Thus, we observe a degeneration of the estimation quality for targets with velocity $v_3 = 4m/s$. A recovery process should be setup here to deal with this problem. The estimation error of DKF_LSR is presented in figure 13. As we can see, this method converges also for different target speeds but in a slow rate compared with CKF, because the state vector and the covariance matrix are exchanged between large and large sets of nodes when the simulation progresses.

We show in figure 13 the convergence speed of the different simulated methods. We see that the estimation error of all the methods converges to zero but with different rates. CKF converges with high speed than the other methods.

VI. Conclusion and Future Work

The energy problem remains a key issue for emerging WSN applications such as target tracking. Our distributed Kalman filtering method coupled with dynamic clustering protocol

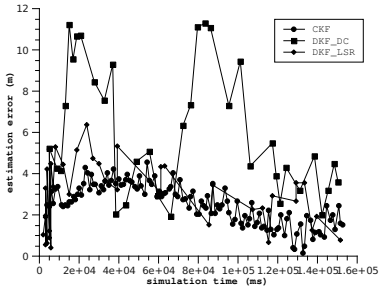

Fig. 13. Estimation quality of the different schemes

(DKF_DC) helped at reducing the network energy consumption in WSN with limited sensing range. We prevented nodes from waking up periodically and limited the selection process within the area close to the target which we organize as a cluster. We manage then this cluster to follow-up the target trajectory. We improved the energy efficiency of the network but the estimation quality has slightly degraded. Including the sensing capabilities of nodes into the selection process and integrating a recovery process when losing targets with complex state model appear as the most promising track for future work.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[3] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*. IEEE Computer Society, Dec. 2005, pp. 8179–8184.

[4] D. Spanos, R. Olfati-Saber, and R. Murray, "Dynamic consensus on mobile networks," in *The 16th IFAC World Congress, Prague, Czech*, 2005.

[5] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[6] U. A. Khan and J. M. F. Moura, "Distributed kalman filters in sensor networks: Bipartite fusion graph," *SSP*, 2007.

[7] S. Kar, "Large scale networked dynamical systems: Distributed inference," Ph.D. dissertation, Carnegie Mellon University Pittsburgh, PA, May 2010.

[8] R. Olfati-Saber and N. F. Sandell, "Distributed tracking in sensor networks with limited sensing range," in *American Control Conference, 2008*. IEEE, Jun. 2008, pp. 3157–3162.

[9] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, "Soi-kf: Distributed kalman filtering with low-cost communications using the sign of innovations," *IEEE Transactions on Signal processing*, vol. 54, no. 12, pp. 4782–4795, Dec. 2006.

[10] H. Medeiros, J. Park, and A. C. Kak, "Distributed object tracking using a cluster-based kalman filter in wireless camera networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 02, no. 04, pp. 448–463, Aug. 2008.

[11] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137.

[12] G. Welch and G. Bishop, "An introduction to the kalman filter," *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 1995.

 ISBN: 978-1-61208-166-3